VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

# BAKALÁŘSKÁ PRÁCE

Brno, 2020 Jakub Sohr


# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS


# WEBOVÁ VIZUALIZACE ANONYMNÍHO POVĚŘOVACÍHO SCHÉMATU
WEB-BASED VISUALIZATION OF ANONYMOUS CREDENTIAL SCHEME

**BAKALÁŘSKÁ PRÁCE**
BACHELOR'S THESIS


**AUTOR PRÁCE** — **Jakub Sohr**
AUTHOR

**VEDOUCÍ PRÁCE** — **Ing. Petr Dzurenda, Ph.D.**
SUPERVISOR


**BRNO 2020**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

# Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

***Student:*** Jakub Sohr ***ID:*** 198044
***Ročník:*** 3 ***Akademický rok:*** 2019/20

**NÁZEV TÉMATU:**

## Webová vizualizace anonymního pověřovacího schématu

**POKYNY PRO VYPRACOVÁNÍ:**

Téma práce je zaměřeno na implementaci webové aplikace pro vizualizaci moderních anonymních pověřovacích schémat. Výstupem práce bude funkční webová aplikace (HTML5, Flash, Java, apod.) demonstrující princip fungování zadaného anonymního pověřovacího schématu.

**DOPORUČENÁ LITERATURA:**

[1] CAMENISCH, J.; DRIJVERS, M.; HAJNÝ, J. Scalable Revocation Scheme for Anonymous Credentials Based on n-times Unlinkable Proofs. In WPES '16 Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society. NY, USA: ACM New York, 2016. p. 123-133. ISBN: 978-1-4503-4569-9.

[2] HAJNÝ, J.; DZURENDA, P.; CAMENISCH, J.; DRIJVERS, M. Fast Keyed-Verification Anonymous Credentials on Standard Smart Cards. In ICT Systems Security and Privacy Protection. Springer Nature Switzerland, 2019. p. 1-13. ISBN: 978-3-030-22312-0.

***Termín zadání:*** 3.2.2020 ***Termín odevzdání:*** 8.6.2020

***Vedoucí práce:*** Ing. Petr Dzurenda, Ph.D.

**doc. Ing. Jan Hajný, Ph.D.**
předseda rady studijního programu

## ABSTRAKT

Tato práce se zabývá systémy pro atributovou autentizaci a kryptografickými primitivy tvořící tyto systémy. Hlavním zaměřením této práce je schéma RKVAC, které je implementováno i vizualizováno. Práce popisuje entity a protokoly vyskytující se v schématu RKVAC. Výstupem této práce je webová aplikace, která slouží pro představení schématu RKVAC širší veřejnosti.

## KLÍČOVÁ SLOVA

RKVAC, KVAC, wBB, Vue, JavaScript, vizualizace, revokace, anonymní pověření, schéma ABC, atributová autentizace, MCL

## ABSTRACT

This thesis explores attribute based credential schemes and the cryptographic preliminaries that are the building blocks of such schemes. The main focus of this thesis is the RKVAC scheme, which is implemented and visualized. This thesis also describes the entities and protocols of the RKVAC scheme and their roles within it. The implemented web application is designed to introduce the RKVAC scheme to the public.

## KEYWORDS

RKVAC, KVAC, wBB, Vue, JavaScript, visualization, revocation, anonymous credentials, ABC schemes, attribute based credentials, MCL

SOHR, Jakub. *Webová vizualizace anonymního pověřovacího schématu*. Brno, 2020, 47 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Petr Dzurenda, Ph.D.

# PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Webová vizualizace anonymního pověřovacího schématu“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ..... .....

podpis autora

# PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Petru Dzurendovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Tato práce vznikla jako součást klíčové aktivity KA6 - Individuální výuka a zapojení studentů bakalářských a magisterských studijních programů do výzkumu v rámci projektu OP VVV Vytvoření double-degree doktorského studijního programu Elektronika a informační technologie a vytvoření doktorského studijního programu Informační bezpečnost, reg. č. CZ.02.2.69/0.0/0.0/16_018/0002575.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

Projekt je spolufinancován Evropskou unií.

# Obsah

# Seznam obrázků

# Seznam tabulek

# Seznam výpisů

# Úvod

Procesy autentizace a řízení přístupu jsou nedílnou součástí našich dennodenních životů. Setkáváme se s nimi na internetu v podobě přístupu do internetového bankovnictví, elektronické počty či u sociálních sítí.

V souvislosti s nárůstem internetových služeb, chytrých domácností, IoT atd. roste potřeba na ochranu soukromí, což stávající autentizace na základě identity neposkytuje.

Řešením mohou poskytnou tzv. anonymní pověření, která jsou však složitější na implementaci i na pochopení. Pomocí těchto kryptografických pověření mohou uživatelé anonymně prokázat držení atributů jako je věk, národnost, pohlaví nebo platnost jízdenky. Při celém procesu ověřování je zachována anonymita uživatele, jelikož důkazy držení atributů jsou vzájemně nespojitelné.

Motivací pro vznik této bakalářské práce je snaha přiblížit problematiku anonymních pověření a atributové autentizace širší veřejnosti pomocí vizualizace. Dané téma je obecně náročné pro pochopení z textových zdrojů, vizualizací dochází ke zjednodušení dané problematiky a je tím ulehčeno její pochopení.

Cílem této práce je popis a porozumění kryptografickému protokolu RKVAC, zvolení vhodné kryptografické knihovny a vytvoření aplikace, která bude sloužit pro vizuální demonstraci schématu RKVAC.

Práce rozebírá kryptografická primitiva, na kterých je schéma RKVAC postaveno.

# 1 Teoretická část

V této kapitole jsou popsána teoretická východiska semestrální práce, která následně budou využita v dalších kapitolách práce. V první části se nachází představení kryptografických primitiv. Další podkapitoly se zaměřují na současný stav atributové autentizace a anonymních pověření.

## 1.1 Kryptografická primitiva

Před popisem protokolů, které jsou nedílnou součástí této práce, je nutné nejprve vysvětlit základní mechanismy, na kterých jsou tyto protokoly postaveny. Pojmy představené v této kapitole budou dále využity v dalších částech práce.

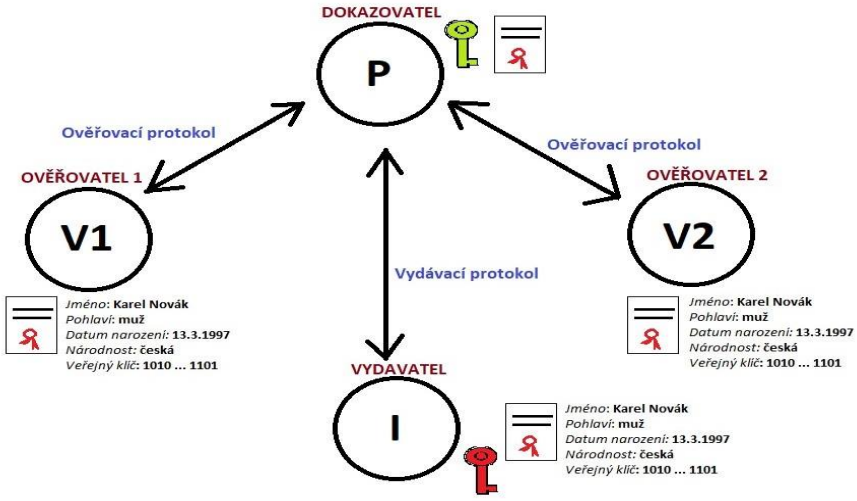
### 1.1.1 Základní pojmy

Mezi základní pojmy spadající do této problematiky patří:

- **Autentizace** – Autentizací se rozumí proces, při kterém se ověřuje identita dané entity.
- **Autorizace** – Vymezení oprávnění daného uživatele v systému.
- **Identifikace** – Proces zjišťování identity uživatele.

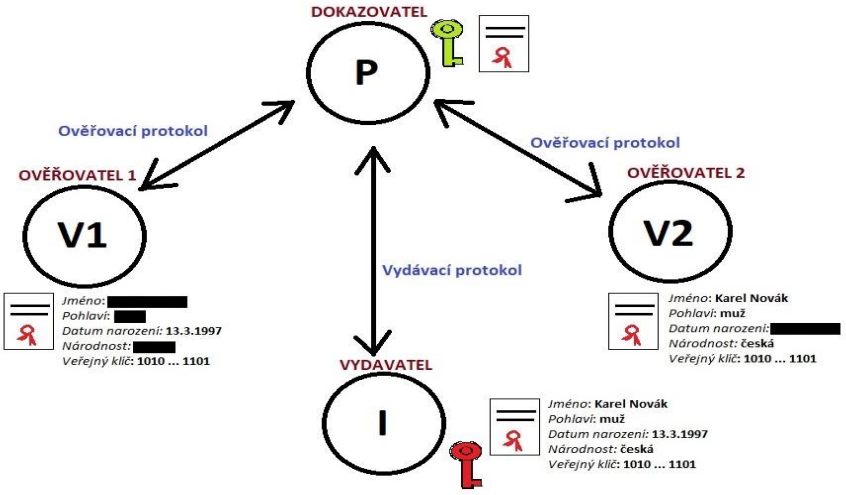
Rozdíl mezi autentizací na základě identity a autentizací na základě atributů spočívá v množství zveřejněných informací. Při autentizaci na základě identity neexistuje jiná možnost, než kompletní zveřejnění identity, což znázorňuje obrázek 1.1. Z něj vyplývá, že uživatel každému ověřovateli pošle všechny informace o své identitě bez ohledu na to, zda ověřovatel tyto informace potřebuje.

Atributová autentizace rozšiřuje toto schéma o možnost zveřejnit pouze podmnožinu atributů uživatelovy identity a zabránit tak úniku citlivých dat. Tato skutečnost je zobrazena na obrázku 1.2, kde uživatel různým ověřovatelům posílá pouze ty atributy, které potřebují.




Obr. 1.1: Autentizace na základě identity

Atributová autentizace je způsob, kterým může entita prokázat některou svoji vlastnost (atribut), aniž by o sobě prozradila příliš mnoho informací. Jako příklad bude uvedena knihovna, kde pro zapůjčení knihy je nutné členství. Při půjčování tedy nemusíme odhalit všechny informace o svojí identitě, stačí pouze prokázat vlastnictví atributu "člen".




Obr. 1.2: Atributová autentizace

V případě nevrácení knihy ve stanoveném čase by měl systém mít možnost zjistit, komu byla kniha zapůjčena a vyvozovat z této situace následky. K tomuto účelu slouží **revokace**, která může mít více podob:

- Revokace autentizačních tokenů – tokeny jsou v rámci systému znehodnoceny, nedají se tedy použít v procesu ověření.
- Revokace nespojitelnosti relací – dojde k odkrytí aktivity daného uživatele, jeho identita však stále zůstává anonymní.
- Revokace anonymity – odhalení identity daného uživatele.

Ideální systém pro anonymní atributová pověření by měl mít následující vlastnosti:

- **Selektivní odhalení atributů** – uživatel má možnost sám zvolit atributy, které odhalí.
- **Anonymita** – identita uživatele není odhalena během procesu autentizace.
- **Nespojitelnost relací** – díky této vlastnosti je profilování uživatele znemožněno.
- **Nesledovatelnost** – implikuje nemožnost vysledování vydavatelem pověření, zda-li požadavek na ověření přišel od stejného zdroje.
- **Revokace** – umožňuje zneplatnit pověření uživatele v systému při porušení pravidel.

Obecně lze říct, že v systémech atributové autentizace vystupují následující entity:

- **Uživatel** – Anonymně prokazuje ověřovateli držení atributů.
- **Ověřovatel** – Ověřuje uživatele, zda-li má přístup ke službě, tj. vlastní potřebné atributy.
- **Vydavatel** – Slučuje atributy do jednoho celku, který podepíše svým soukromým klíčem. Tímto vzniká digitální pověření.
- **Revokační autorita** – Umožňuje revokaci vydaných pověření a identifikaci uživatelů sytému ve spolupráci s ostatními entitami.

Vlastnosti a entity byly převzaty z [1].

### 1.1.2 Protokoly s nulovou znalostí

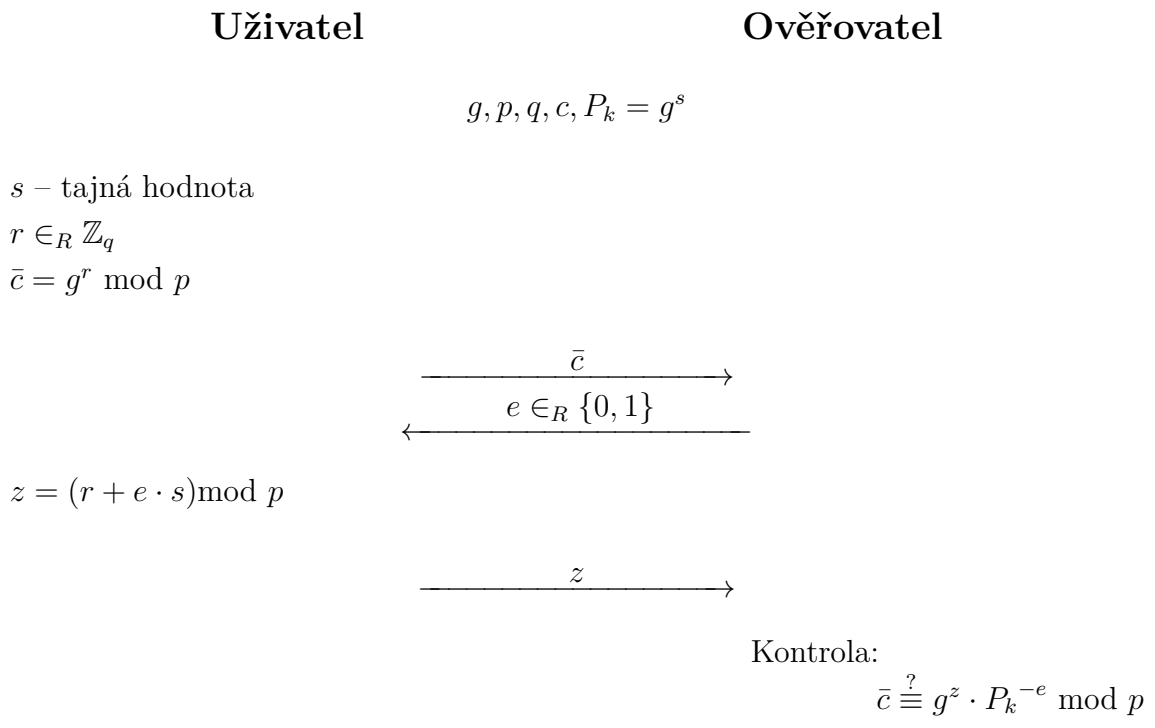
Název vychází z anglického "Zero-Knowledge', tedy protokoly s nulovou znalostí. Jedná se o protokoly, ve kterých vystupují dvě strany - Ověřovatel a Dokazovatel. Dokazovatel se snaží ověřovateli dokázat, že zná tajnou informaci. Ověřovatel ověřuje, zda-li dokazovatel opravdu zná tajnou informaci. Tyto protokoly splňují následující vlastnosti dle [2]:

- **Úplnost** – Poctivý uživatel nebude systémem odmítnut.
- **Spolehlivost** – Nepoctivý uživatel bude vždy odhalen.
- **Nulová znalost** – V průběhu ověřování nikdy není zveřejněna tajná informace, pouze důkaz o její znalosti.

**Příklad.** *Princip tohoto protokolu je ilustrován příkladem dvou kamarádů, z nichž jeden je barvoslepý (Adam) a druhý není (Bob). Dohromady mají dva míče, jeden zelený a jeden červený, lišící se pouze barvou. Pro Adama se tedy míče zdají být*

*stejné. Aby mu ale Bob dokázal, že jsou jiné, navrhne tento systém - Adam si oba míče schová za záda, poté jeden z nich náhodně vytáhne a ukáže jej Bobovi a míč poté opět schová za záda, náhodně jeden vytáhne a zeptá se Boba, jestli ukazuje stejný míč, nebo jiný. Tento proces opakují podle potřeby. Bob dokáže pokaždé odpovědět správně, protože míč pozná podle barvy.* Tímto systém splňuje všechny vlastnosti protokolu s nulovou znalostí.:

- **Úplnost** – Adam bude po dostatečném množství opakovaní přesvědčen, že míče mají různé barvy.
- **Spolehlivost** – pravděpodobnost, že by Bob byl schopný odpovědět na všechny výzvy, kdyby neznal tajnou informaci, je zanedbatelná.
- **Nulová znalost** – Adam, kromě informace, že míče mají jinou barvu, nezná žádnou jinou informaci.




Obr. 1.3: Schnorrův protokol nulové znalosti.

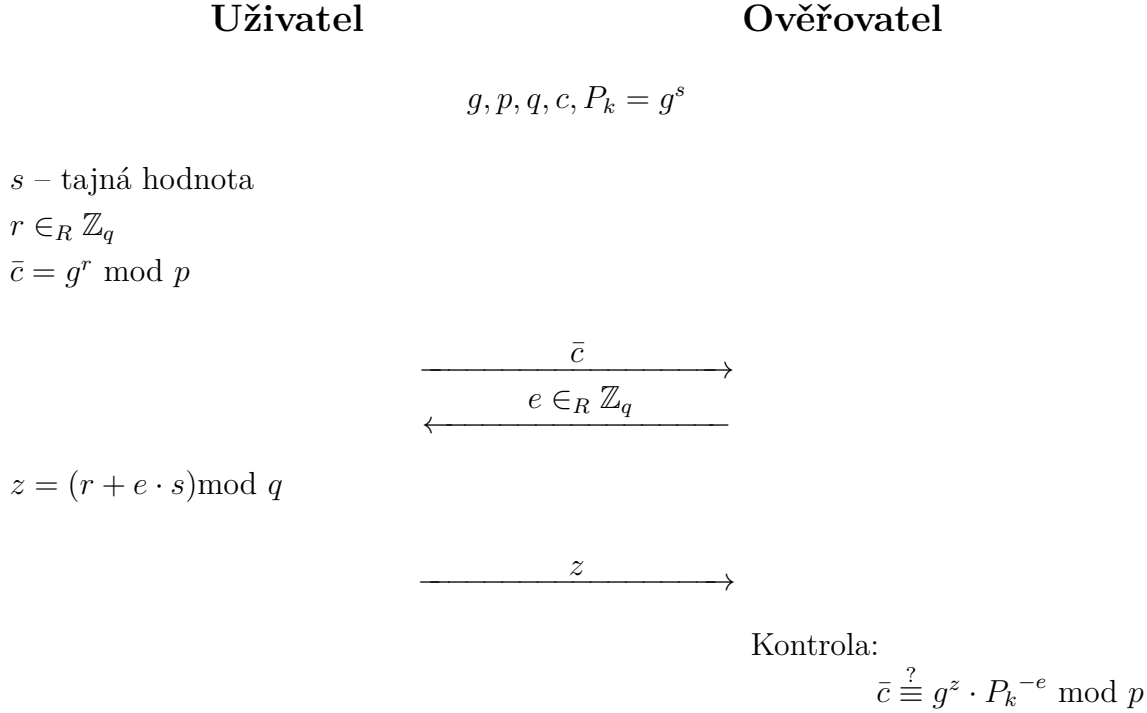
Na obrázku 1.3 je zobrazen průběh Shnorrova protokolu nulové znalosti[3]. V daném protokolu se nejprve uživatel zaváže k hodnotě $\bar{c}$ a následně obdrží od ověřovatele výzvu $e$, která nabývá hodnot 0 nebo 1. Následně dopočítá odpověď a pošle ji ke kontrole. Při každé iteraci tohoto protokolu má útočník 50% šanci, že odpoví správně. Bezpečnostním parametrem tohoto systému je tedy počet iterací. Pravděpodobnost, že by útočník nebyl odhalen protokolem s nulovou znalostí po $n$ pokusech, je možné vypočítat jako $(\frac{1}{2})^n$.

### 1.1.3 Σ-protokoly

V praxi často nahrazují protokoly s nulovou znalostí, jelikož jsou mnohem rychlejší a efektivnější v tom, že jsou pouze třícestné. U protokolu nulových znalostí je nutné pro dokázání znalosti provádět iterace důkazu, protože po dokončení každé iterace je pravděpodobnost, že by útočník byl schopen podvrhnout odpověď 0,5. U Σ-protokolů je bezpečnostním parametrem řád grupy $q$ a pravděpodobnost, že by útočník nebyl odhalen je $2^{-q}$. Příkladem Σ-protokolu je Schnorrův Σ-protokol znázorněn na obrázku 1.4. Σ-protokoly se skládají ze tří částí dle [4]:

- **Závazek** – Uživatel se zaváže k tajné hodnotě.
- **Výzva** – Ověřovatel pošle uživateli výzvu velikosti $k$, kde $k$ představuje bezpečnostní parametr systému.
- **Odpověď** – Uživatel vypočítá důkaz znalosti tajné hodnoty a pošle odpověď ověřovateli.

Tyto protokoly vykazují stejné vlastnosti jako protokoly s nulovou znalostí.




Obr. 1.4: Schnorrův Σ-protokol

### 1.1.4 Bilineární párování

Bilineární mapa může být vytvořena nad eliptickými křivkami. Každá operace pro výpočet $e(P, Q)$ je operace párování. Nechť $G_1$,$G_2$ jsou cyklické aditivní grupy, a

$G_T$ cyklická multiplikativní grupa. Obě grupy $G$ i $G_T$ mají stejný prvočíselný řád $q$. Mapování $e : G_1 \times G_2 \to G_T$ se nazývá bilineárním, pokud splňuje následující podmínky[5]:

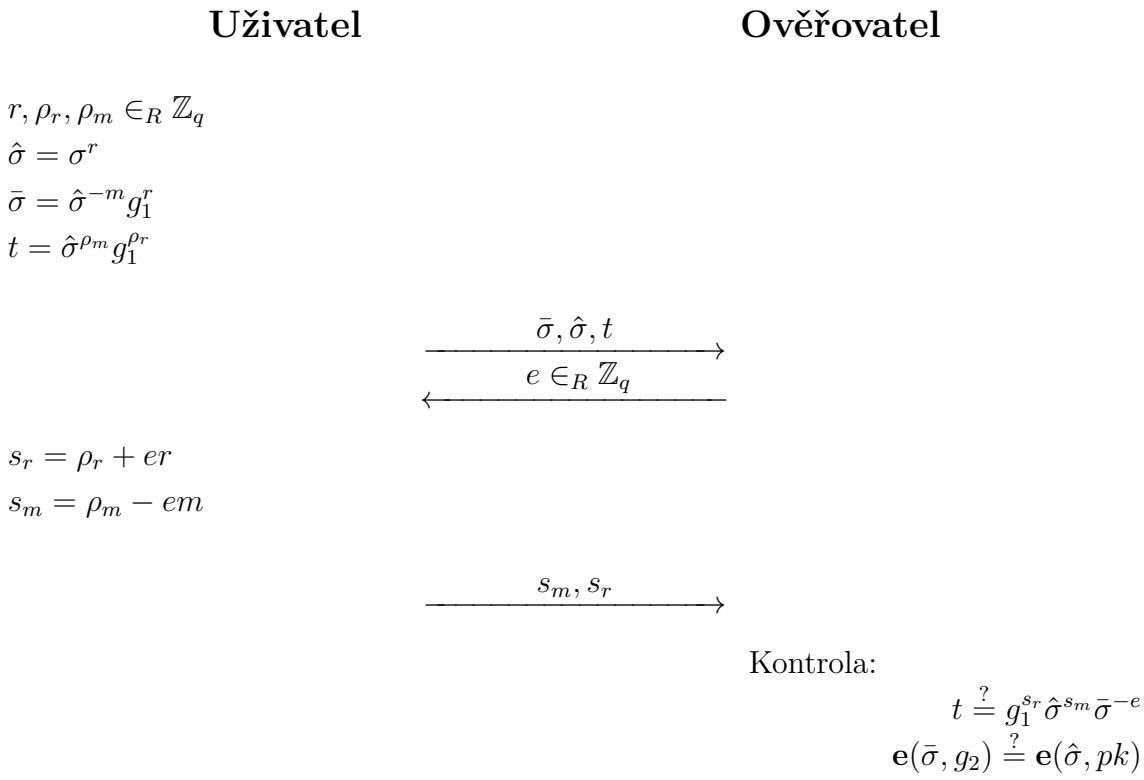
1. bilinearita – $\forall P \in G_1, \forall Q \in G_2, \forall a, b \in \mathbb{Z}_p$, $e(aP, bQ) = e(P, bQ)^a = e(aP, Q)^b = e(P, Q)^{ab}$
2. nedegenerativnost – $\forall P \in G_1, Q \in G_2$ platí, že $e(P, Q) \in G_T$
3. spočitatelnost – existuje algoritmus, který je schopný vypočítat $e(P, Q)$ pro všechny $P \in G_1, Q \in G_2$.

### 1.1.5 Weak Boneh-Boyen signature

Weak Boneh-Boyen (wBB) je podpisové schéma založené na předpokladu q-SDH viz [6, 7]. Protokol definuje tři algoritmy, které jsou nepostradatelné pro jeho správný průběh. Mezi tyto algoritmy jsou řazeny **Setup**, **Sign** a **Verify**.

- **Setup** – tímto algoritmem jsou inicializovány grupy $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ a jejich příslušné generátory $g_1, g_2$. Také vytvoří bilinearní mapu $e$ a vygeneruje soukromý a veřejný klíč.
- **Sign** – algoritmus pro vytváření podpisu, na základě zprávy a soukromého klíče vytvoří podpis $\sigma = g_1^{\frac{1}{sk+m}}$.
- **Verify** – Tento algoritmus ověřuje podpis $\sigma$, nezajišťuje však vlastnost nespojitelnosti relací, jelikož vyžaduje zprávu (atribut) pro ověření pravosti podpisu.

Na obrázku 1.5 je zobrazen průběh důkazu znalosti wBB podpisu. V kontextu tohoto důkazu je $\sigma$ podpis získaný protokolem Sign a $m$ je daný atribut. Nejprve jsou uživatelem náhodně zvoleny tři prvky z $\mathbb{Z}_q$. Poté uživatel podpis znáhodní a zaváže se k hodnotám $\hat{\sigma}, \bar{\sigma}, t$ a obdrží výzvu $e$ od ověřovatele. Na tuto výzvu sestrojí odpověď, tedy důkaz znalosti tajné hodnoty, kterou odešle ověřovateli k ověření, ten výzvu ověří a rozhodne, zda-li je uživateli přístup umožněn či odepřen. Díky těmto náhodným hodnotám není při ověřování odhalen ani atribut ani podpis, čímž schéma zajišťuje nespojitelnost relací. Tento protokol je základním pilířem vizualizovaného protokolu RKVAC[8]. Průběh důkazu znalosti randomizovaného wBB podpisu je znázorněn na obrázku 1.5.




Obr. 1.5: Důkaz znalosti randomizovaného wBB podpisu.

## 1.2 Atributová autentizace

Existuje mnoho druhů systémů, které implementují atributovou autentizaci. Různorodost vychází z odlišných vlastností, kterými systém disponuje. V některých systémech také může chybět entita revokační autority 1.1.1. V následující tabulce 1.1 jsou ilustrovány rozdíly mezi jednotlivými současnými systémy, informace o systémech Idemix a U-Prove převzaty z [1].

Tab. 1.1: Porovnání současných systémů.

| | Idemix | U-Prove | RKVAC |
|---|---|---|---|
| Ověřování pomocí vybraných atributů | ✓ | ✓ | ✓ |
| Anonymita | ✓ | ✓ | ✓ |
| Bezpečnost | RSA | DL | q-SDH |
| Nespojitelnost relací | ✓ | ✗ | ✓ |
| Nepřenositelnost | ✓ | ✓ | ✓ |
| Nesledovatelnost | ✓ | ✓ | ✓ |
| Možnost revokace atributů | ✓ | ✗ | ✓ |
| Možnost odhalení útočníka | ✗ | ✗ | ✓ |
| Počet operací modulárního mocnění | u+1 | u+3 | u+2 |

### 1.2.1 U-Prove

U-Prove je kryptografický systém, navržen firmou Microsoft, implementující atributovou autentizaci. Základem systému U-Prove je U-Prove token, tím se rozumí kryptograficky zabezpečený soubor jakýchkoliv informací označovaných jako atributy. Tyto tokeny vydává autoritativní zdroj pomocí vydávacího protokolu, a později jsou prezentovány ověřovateli pomocí ověřovacího protokolu. Jelikož je U-Prove token pouze blok binárních dat, je možné jej vydávat i ověřovat přes jakoukoliv elektronickou síť. Pro účast v U-Prove protokolech potřebují všechny zúčastněné strany své výpočetní zařízení.

#### Vydávání U-Prove tokenu

Aby dokazovatel mohl dostat U-Prove token od vydavatele, musí být obě strany zapojeny do vydávacího protokolu. Tento protokol bere na vstupech, mimo jiné, atributy, které mají být zapsány do tokenu. Jelikož je při vydáváni použit Schnorrův zaslepený podpis, ani sám vydavatel nezná obsah tokenu. Vydávací protokol umožňuje vydavateli zabezpečit U-Prove tokeny proti neautorizovaným zásahům. Následující vlastnosti jsou vždy zaručeny:

### 1.2.2 Identity Mixer

Identity Mixer je atributově autentizační schéma vyvinuté společností IBM v Curychu. Bezpečnost tohoto schématu závisí na složitosti faktorizace RSA. Tento systém poskytuje nespojitelnost relací. Revokace pověření funguje na bázi období platnosti těchto pověření, kdy v případě porušení podmínek není pověření prodlouženo.

Přehled systému Identity Mix je zobrazen na obrázku 1.6. Je zde znázorněn základní princip systému Identity Mixer.

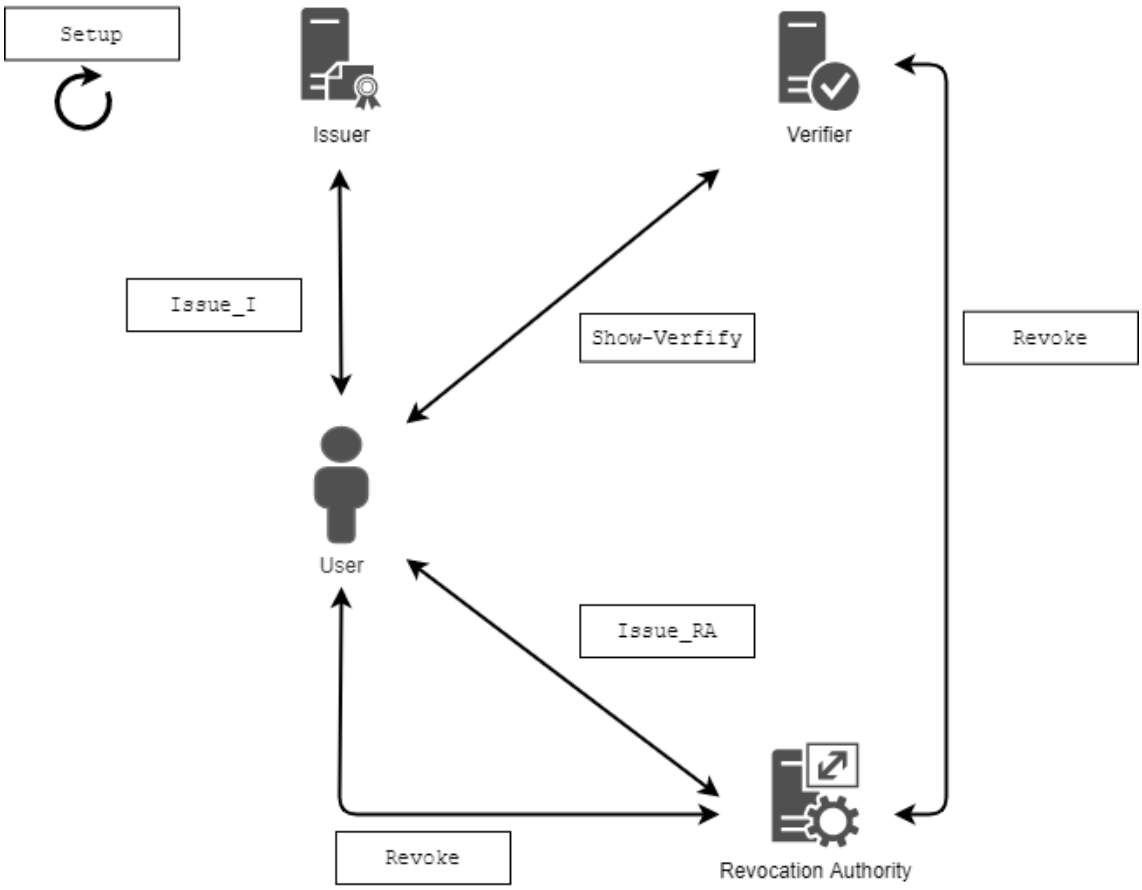

Obr. 1.6: Přehled systému Identity Mixer

### 1.2.3 RKVAC

Revocable Keyed-Verification Anonymous Credential(RKVAC) scheme, je atributově autentizační schéma využívající faktu, že vydavatel a ověřovatel sdílí soukromý klíč. Díky tomu je možné při ověřování uživatele použít symetrickou kryptografii. Princip symetrické kryptografie, na kterém je založen algebraický autentizační kód zprávy $\text{MAC}_{\text{wBB}}$ viz [8, 9]. Hlavním kryptografickým primitivem tohoto schématu je podpis wBB. Na obrázku 1.7 je znázorněna architektura schématu RKVAC. Můžeme zde vidět entity, které v schématu vystupují, a také protokoly, které mezi nimi probíhají.

#### Entity vystupující v schématu RKVAC

**Uživatel** – Při registraci obdrží vydavatelem podepsané pověření, které obsahuje revokační atribut podepsaný revokační autoritou a atributy uživatele. Každému uživateli je také přiděleno jednoznačně identifikující ID, díky kterému je možno zjistit identitu uživatele, poruší-li pravidla. Toto ID je svázáno s revokačním handlerem při procesu vydávání revokačního handleru revokační autoritou.
**Vydavatel** – Úkolem vydavatele je vydávání kryptografických pověření uživatelům, poskytnou-li revokační atribut.
**Ověřovatel** – Kontroluje validitu uživatelova pověření a zároveň kontroluje, zda-li uživatel není na revokační listině. Na základě těchto kritérií uživateli povolí nebo nepovolí přístup. V případě, že uživatel poruší pravidla, žádá revokační autoritu o revokaci nebo identifikaci uživatele, podle závažnosti porušení.




Obr. 1.7: Architektura schéma RKVAC

**Revokační autorita** – Vydává revokační atributy uživatelům a zároveň udržuje revokační listinu. Způsob dělení entit převzat z [10].

### Protokoly schématu RKVAC

**Setup_I**

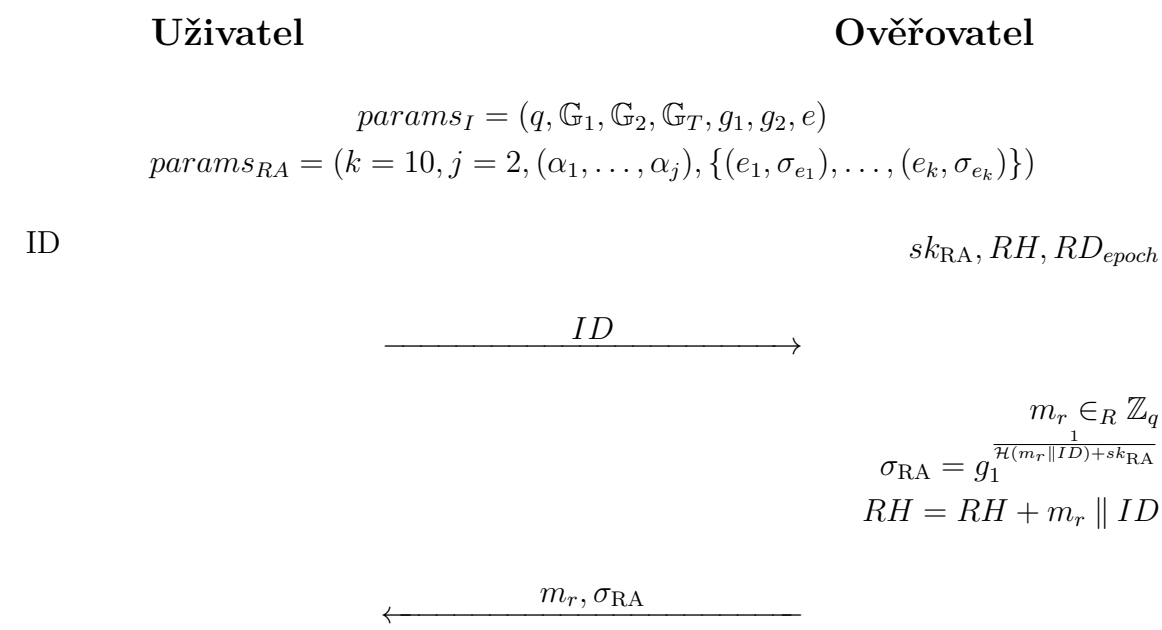
Tento protokol zajišťuje inicializaci systému pro další funkčnost. Vygeneruje veřejné parametry, které jsou dále implicitním vstupem všech protokolů. Tedy $\text{params}_I = (x_r, x_0, x_1, \ldots, x_{n-1})$, kde $n$ je počet atributů.

**Setup_RA**

Tento protokol přebírá parametry z protokolu Setup_I. Revokační autorita náhodně zvolí svůj soukromý $sk_{\text{RA}}$ a z něj vypočítá veřejný klíč jako $pk_{\text{RA}} = g_2^{sk_{\text{RA}}}$. Dále inicializuje revokační listinu RL.

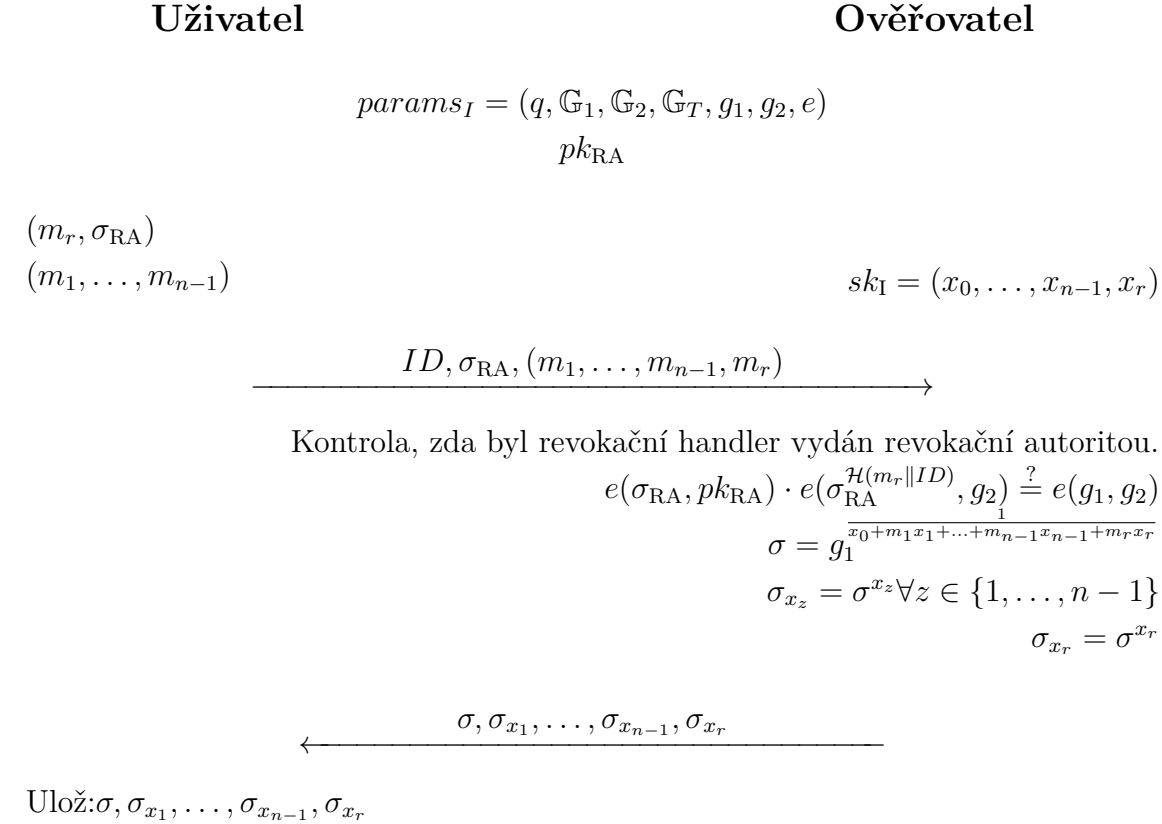
### Vydávací protokol

Vydávací protokol se skládá ze dvou částí. Uživatel musí nejprve získat relokační atribut, který je svázán s jeho ID. Poté může žádat vydavatele o vytvoření pověření. **Issue_RA** – Revokační autorita náhodně zvolí hodnotu revokačního atributu $m_r$. A poté vypočte podpis příslušící k danému $m_r$ a ID uživatele jako $\sigma_{RA} = \mathrm{g_1}^{(\mathcal{H}(m_r||\mathrm{ID})+\mathrm{sk_{RA}})^{-1}}$. Interně si revokační autorita vytvoří záznam o uživatelově revokačním atributu a jeho ID do své databáze.




Obr. 1.8: Průběh protokolu Issue_RA

**Issue_I** – Na základě vydavatelova soukromého klíče, revokačního atributu uživatele $m_r$,odpovídajícího podpisu revokační autority $\sigma_{RA}$ a pole atributů $m_1, \ldots, m_n$ vydá uživateli pověření.




Obr. 1.9: Průběh protokolu Issue_I

### Ověřovací protokol

V ověřovacím protokolu dochází k interakci mezi uživatelem a ověřovatelem. Uživatel nejdříve obdrží od ověřovatele údaj o epoše spolu s náhodně vygenerovaným číslem pro jedno použití (*nonce*). Poté si uživatel může vybrat, které atributy zveřejní a které ne. Poté sestaví důkaz znalosti nezveřejněných atributů a vypočte pseudonym. Ověřovateli pak předává důkaz znalosti nezveřejněných atributů, pseudonym a zveřejněné atributy. Ověřovatel poté zkontroluje, zda je pověření platné, revokované nebo podvržené. Průběh protokolu je znázorněn na obrázku 1.10.

**Uživatel** **Ověřovatel**

$\text{sk}=(x_0, \ldots, x_{n-1}, x_r)$

$\xleftarrow{nonce, epoch}$

$params_{RA} = (k = 10, j = 2, (\alpha_1, \ldots, \alpha_j), \{(e_1, \sigma_{e_1}, \ldots, (e_k, \sigma_{e_k})\})$

$(m_1, \ldots, m_{n-1}, m_r)$

$\sigma, (\sigma_{x_1}, \ldots, \sigma_{x_{n-1}}, \sigma_{x_r})$

$e_I, e_{II} \in_R (e_1, \ldots, e_k)$

$\sigma_{e_I}, \sigma_{e_{II}} \in_R (\sigma_{e_1}, \ldots, \sigma_{e_k})$

$i = \alpha_1 e_I + \alpha_2 e_{II}$

$\text{C}= g_1^{(i-m_r+\mathcal{H}(epoch))^{-1}}$

$\rho, \rho_v, \rho_i, \rho_{m_r}, \rho_{m_{z\in D}}, \rho_{eI}, \rho_{eII} \in_R \mathbb{Z}_q$

$\hat{\sigma} = \sigma^{\rho}$

$\hat{\sigma}_{e_I} = \sigma^{\rho_{e_I}}, \hat{\sigma}_{e_{II}} = \sigma^{\rho_{e_{II}}}$

$\bar{\sigma}_{e_I} = \hat{\sigma}_{e_I}^{-e_I} g_1^{\rho}, \bar{\sigma}_{e_{II}} = \hat{\sigma}_{e_{II}}^{-e_{II}} g_1^{\rho}$

$t_{\text{verify}} = g_1^{\rho_v} \sigma_{x_r}^{\rho_{mr}\cdot\rho} \prod_{z\notin D} \sigma_{x_z}^{\rho_{mz}\cdot\rho}$

$t_{\text{revoke}} = C^{\rho_{mr}} C^{\rho_i}$

$t_{\text{sig}} = g_1^{\rho_i} h_1^{\rho_{e_I}} h_2^{\rho_{e_{II}}}$

$t_{\text{sigI}} = g_1^{\rho_v} \hat{\sigma}_{e_I}^{\rho_{e_I}}, t_{\text{sigII}} = g_1^{\rho_v} \hat{\sigma}_{e_{II}}^{\rho_{e_{II}}}$

$e = \mathcal{H}(t_{\text{verify}}, t_{\text{revoke}}, t_{\text{sig}}, t_{\text{sigI}}, t_{\text{sigII}}, \hat{\sigma}, \hat{\sigma}_{e_I}, \hat{\sigma}_{e_I I}, \bar{\sigma}_{e_I}, \bar{\sigma}_{e_{II}}, C, nonce)$

$\langle S_{m_z} = \rho_{m_z} - e m_z \rangle_{z\notin D}$

$S_v = \rho_v + e\rho$

$S_{m_r} = \rho_{m_r} - e m_r$

$S_i = \rho_i + ei$

$S_{e_I} = \rho_{e_I} - e e_I, S_{e_{II}} = \rho_{e_{II}} - e e_{II}$

$\pi = (e, S_{m_{z\notin D}}, S_v, S_{m_r}, S_i, S_{e_I}, S_{e_I I})$

$\xrightarrow{m_{z\in D}, \pi, C, \hat{\sigma}, \hat{\sigma}_{e_I}, \hat{\sigma}_{e_I I}, \bar{\sigma}_{e_I}, \bar{\sigma}_{e_{II}}}$

$t_{\text{verify}} = \hat{\sigma}^{e x_0} \cdot g_1^{s_v} \cdot \prod_{z\notin D} \hat{\sigma}^{x_z s_{m_z}} \cdot \prod_{z\in D} \hat{\sigma}^{-e x_z m_z}$

$t_{\text{revoke}} = \left(g_1 C^{-\mathcal{H}(epoch)}\right)^{-e} C^{S_{m_r}} C^{s_i}$

$t_{sig} = g_1^{s_i} h_1^{s_{e_I}} h_2^{s_{e_I I}}$

$t_{sigI} = \bar{\sigma}_{eI}^{-e} \hat{\sigma}_{e_I}^{S_{e_I}} g_1^{S_v}$

$t_{sigII} = \bar{\sigma}_{e_{II}}^{-e} \hat{\sigma}_{e_{II}}^{S_{e_{II}}} g_1^{S_v}$

$e_{\text{verifier}} = \mathcal{H}(t_{\text{verify}}, t_{\text{revoke}}, t_{\text{sig}}, t_{\text{sigI}}, t_{\text{sigII}}, \hat{\sigma}, \hat{\sigma}_{e_I}, \hat{\sigma}_{e_I I}, \bar{\sigma}_{e_I}, \bar{\sigma}_{e_{II}}, C, nonce)$

Kontrola:

$e \stackrel{?}{=} e_{\text{verifier}}$

$\mathbf{e}(\bar{\sigma}_{e_I}, g_2) \stackrel{?}{=} \mathbf{e}(\hat{\sigma}_{e_I}, pk_{\text{RA}})$

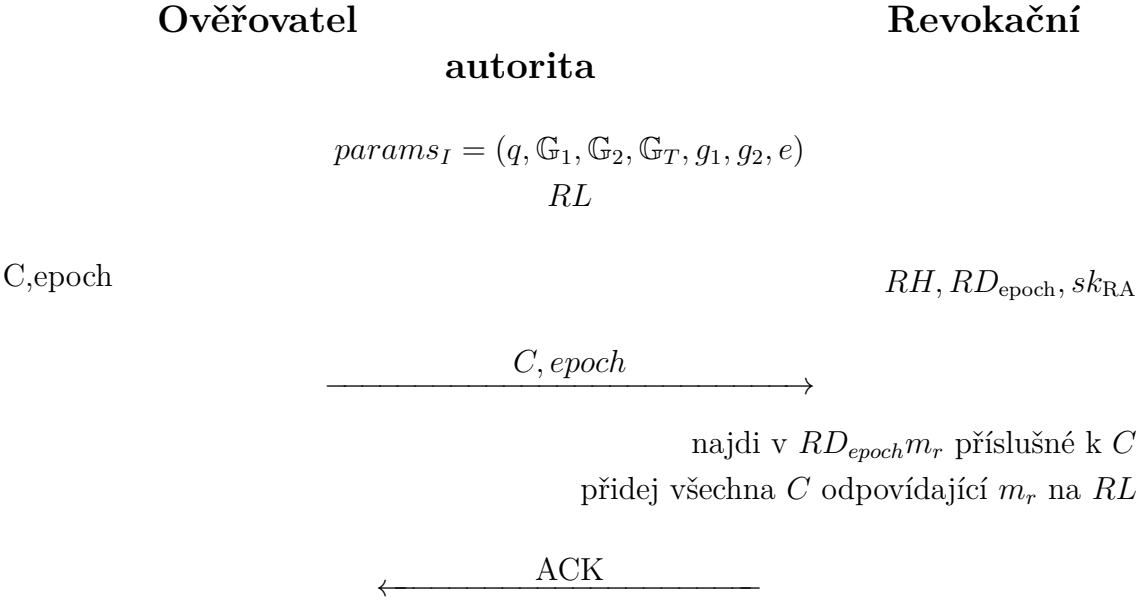
$\mathbf{e}(\bar{\sigma}_{e_{II}}, g_2) \stackrel{?}{=} \mathbf{e}(\hat{\sigma}_{e_{II}}, pk_{\text{RA}})$

$C \notin \text{RL}$

Obr. 1.10: Průběh ověřovacího protokolu

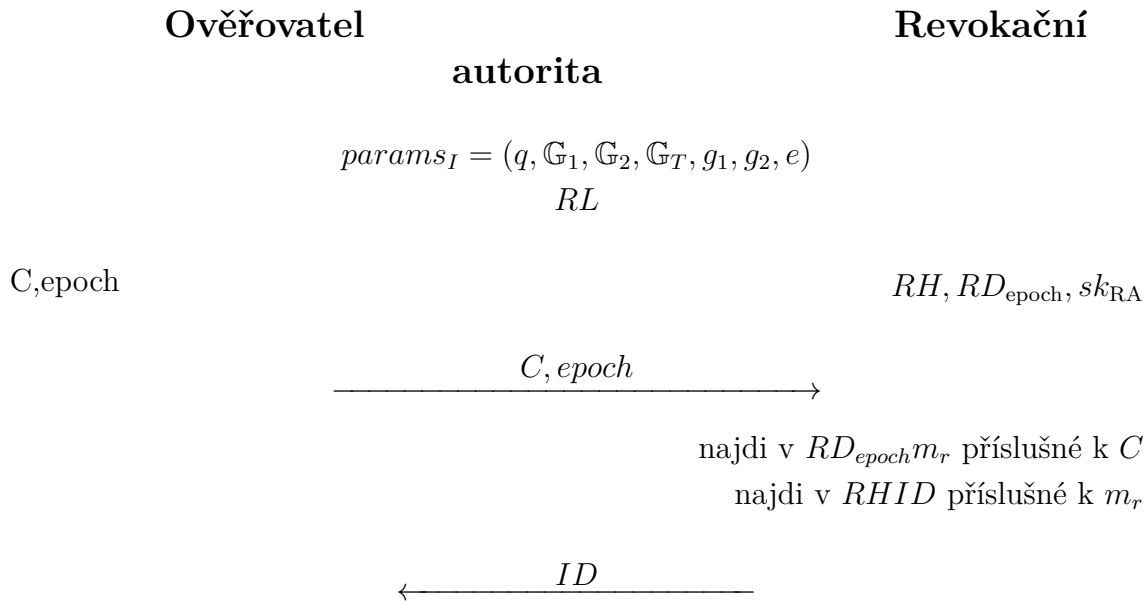
### Revokační protokol

Revokační protokol poskytuje možnost zneplatnit již vydané pověření. Protokol funguje díky veřejné revokační listině ($RL$). Tuto listinu spravuje revokační autorita. Na této listině se vyskytují revokované pseudonymy dané epochy. Při každém ověřování ověřovatel provádí kontrolu, zda-li není pseudonym uživatele na revokační listině viz 1.10.




Obr. 1.11: Průběh revokačního protokolu

Revokační autorita také poskytuje možnost revokovat anonymitu uživatele a tím jej identifikovat. K tomu slouží identifikační protokol. Aby ale protokol fungoval, musí si revokační autorita vést seznam vydaných revokačních handlerů $m_r$ a k nim příslušných ID. Tento seznam se označuje $RH$. Průběh protokolu je demonstrován na obrázku 1.12




Obr. 1.12: Průběh identifikačního protokolu

# 2 Použité technologie

V této kapitole jsou představeny technologie potřebné k implementaci funkční webové aplikace, která slouží k vizualizaci schématu anonymních pověření. Dále je zde zvolena a popsána kryptografická knihovna.

## 2.1 HTML

HTML (HyperText Markup Language), je hypertextový značkovací jazyk sloužící k reprezentaci webové stránky v textové podobě a je následně interpretován webovým prohlížečem. HTML byl vytvořen Tim Berners-Lee a Robert Cailliau roku 1990. Od této doby prošel jazyk mnoha změnami, kdy v současnosti je aktuální verze 5.

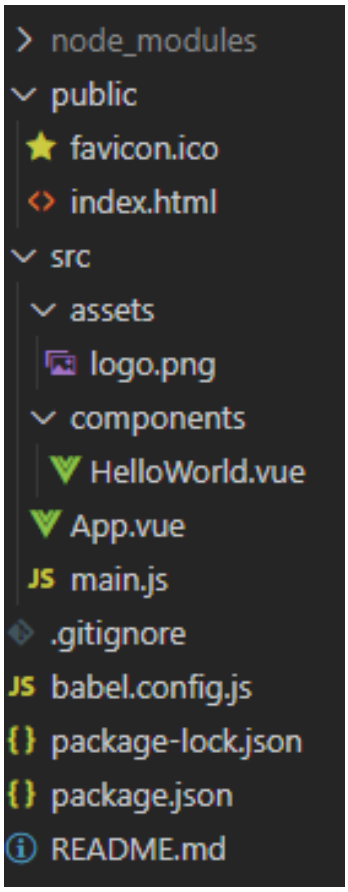
Pro vytváření a formátování webových stránek využívá předdefinovaných značek, tzv. elementů. Ty se dále skládají z tagů a atributů. Základní struktura HTML dokumentů se skládá z tzv. head – hlavičky a body – těla. V hlavičce se zpravidla nacházejí informace o kódování dokumentu, titulek stránky a podobně. Tělo obsahuje samotný obsah webové stránky. Příklad zdrojového kódu z aktuální verze 5 je zobrazen na výpisku 2.1 .

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <script src="skript.js"></script>
5      <title>Nadpis stránky</title>
6    </head>
7    <body>
8      <p>Obsah webové stránky</p>
9    </body>
10 </html>
```

Výpis 2.1: Příklad HTML

HTML 5 je nejnovější verzí tohoto hypertextového značkovacího jazyka. Přechozí verze 4 byla vydána roku 1997 a následně byla lehce opravena roku 1999. Poté následovala pauza, po kterou byl vývoj tohoto jazyka pozastaven z důvodu vývoje nového jazyka XHTML, který se však na trhu neuchytil. Proto v roce 2008 došlo k obnovení vývoje jazyka HTML a v roce 2014 byla vydána verze 5. Informace byly čerpány z [11].

Tato verze odstraňuje zastaralé řídící značky a přidává velké množství nové funkcionality zaměřené na dynamický a multimediální obsah. Mezi tyto nové funkce například patří i canvas a implementace protokolů websocket. Také přidává možnost prohlížení stránek v režimu offline.




Obr. 2.1: Obsah adresáře nově vytvořeného projektu Vue

## 2.2 JavaScript

JavaScript, často zkracován na "JS", je znám jako skriptovací jazyk pro webové aplikace, avšak v dnešní době jej často najdeme i mimo webové prostředí, například v Node.js nebo Adobe Acrobat. JavaScript je "vyšší programovací jazyk" což znamená, že poskytuje vyšší míru abstrakce, kód je tedy pro lidi srozumitelnější než například kód assembleru. Umožňuje objektově-orientované programování na základě funkcí jako konstruktorů objektů nebo dědění pomocí prototypů. Informace byly čerpány z [12].

### 2.2.1 Vue

Vue.js je JavaScriptový framework, který umožňuje rychlý a snadný vývoj jednostránkových webových aplikací. Vue také usnadňuje udržování a rozšiřování díky dělení kódu do jednotlivých komponent.

Pro vytvoření Vue.js projektu je nejprve nutné stáhnout node package manager (npm), dostupný na nodejs.org. Po úspěšné instalaci npm je již možné v příkazovém

řádku pomocí příkazu

```
npm install vue
```

nainstalovat Vue. Nový vue.js projekt je možné vytvořít pomocí příkazu

```
vue create jmeno-projektu
```

Vue poté vytvoří v současném adresáři složku se stejným jménem. Obsah složky je možno vidět na obrázku 2.1.

Vue sjednocuje všechny webové technologie do jednoho souboru. JavaScript, css viz [13] i html jsou tedy v jediném souboru, který je před vydáním kompilován do samostatných standardních souborů. Vue také zajišťuje server pro vývojářské účely. Ten se spouští příkazem

```
npm run serve
```

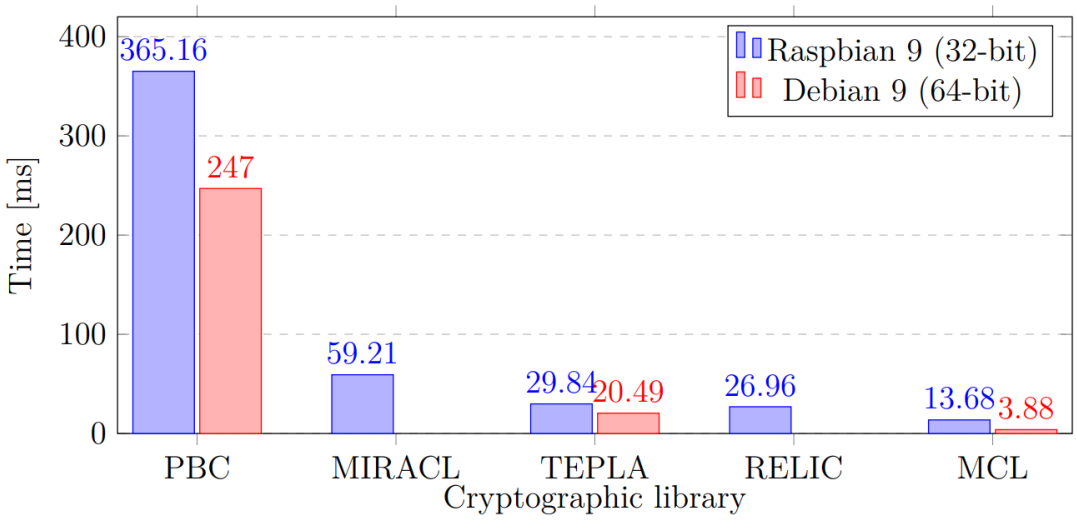
a obnovuje svůj obsah pokaždé, když se uloží změny v některém ze zdrojových souborů. Když je obsah připraven ke zveřejnění, Vue pomocí příkazu

```
npm run build
```

vytvoří minimalizovanou a optimalizovanou verzi webové stránky. Tím zajistí, že se webové stránky načtou rychle i na pomalejším připojení.

### 2.2.2 MCL

MCL je knihovna, jejíž autorem je Shigeo Mitsunari [14]. Tato knihovna poskytuje funkce pro práci s eliptickými křivkami. Obsahuje několik tříd, zejména pro body v grupách $G_1, G_2$ a $G_T$, na kterých definuje sčítání, násobení a unární mínus. Pro potřeby této práce je pak nejzajímavější funkcí bilineární párování.




Obr. 2.2: Porovnání dostupných knihoven

Na obrázku 2.2 je znázorněno porovnání dostupných knihoven poskytujících bilinearní párování. Výsledné časy v grafu byly převzaty z [15].

# 3 Popis implementace

V této kapitole je popsána implementace uživatelského rozhraní webové aplikace a kryptografického jádra.

Moderní vzhled a funkčnost stránek zajišťuje Vuetify[16], což je knihovna designových prvků pro framework Vue. Tuto knihovnu je možno do existujícího Vue projektu přidat příkazem

```
vue add vuetify
```

## 3.1 Kryptografické jádro schématu

Pro implementaci kryptografického jádra schématu byl zvolen jazyk JavaScript. Jelikož integrální součástí schématu KVAC je operace bilineárního párování, bylo nutné do projektu importovat knihovnu MCL, popsána v sekci 2.2.2. Import knihovny MCL byl proveden příkazem

```
npm -i --save mcl-wasm
```

Balíčkovací manager node poté stáhne příslušné moduly do složky `src\node_modules`. Posledním krokem je přidat knihovnu MCL do jmenného prostoru Vue aplikace pomocí příkazů `import mcl from "mcl-wasm"` a `Vue.use(mcl)`. Poté už je možné knihovnu inicializovat pomocí `mcl.init(mcl.BN254)`. Obdobně je nutné importovat bezpečnou hashovací funkci sha256. Bezpečnostně ověřená implementace funkce sha256 se nachází v balíčku "CryptoJS". Ten je stažen pomocí příkazu

```
npm -i --save CryptoJS
```

a poté importován pomocí příkazů `import CryptoJS from "crypto-js"` a `Vue.use(CryptoJS)`.

Kryptografické jádro provádí inicializaci automaticky po otevření stránky uživatelem, kdy je zavolaná funkce `window.onload()`. V rámci inicializace jsou vygenerovány parametry systému, tedy generátory g1 a g2. Dále jsou inicializovány funkce bilineárního párování a počáteční stav systému, tedy instanciování tříd `User`, `Issuer` a `Revocation authority`.

### 3.1.1 Třída User

Třída User definuje prototyp uživatele. Existuje více možností jak definovat třídu v JavaScriptu, nejjednodušší a tedy i zvolenou možností je vytvořit funkci jako objekt. Definice třídy User vypadá následovně:

```
    function User(mcl,g1,CryptoJS) {
    //obsah
```

```
3       }
```

Výpis 3.1: Definice třídy User

Na výpisku 3.1 můžeme vidět, že třída User přebírá jako argument mcl,g1 a CryptoJS, tedy parametry systému, které potřebuje k běhu. Dále tato třída implementuje funkce `Setup`, `getRevocationAttributes`, `getIssuerAttributes`, `Show` - tedy všechny protokoly, kterých se účastní uživatel.

Funkce `Setup` je definována dle výpisku 3.2.

```
1  this.setup = function () {
2      for (let i = 0; i < attributeCount; i++) {
3          this.disclosed.push(true);
4      }
5      //pole dvojic indexů
6      for (let i = 0; i < attributeCount; i++) {
7          for (let j = 0; j < attributeCount; j++) {
8              this.avilableIndexes.push([i, j]);
9          }
10     }
11     //prvni prvek je revokacni atribut- nezverejnitelny
12     this.disclosed[0] = false;
13     for (let i = 0; i < attributeCount; i++) {
14         let mi = new mcl.Fr();
15         this.attributesFr.push(mi);
16     }
17 }
```

Výpis 3.2: Definice funkce Setup

V této funkci se inicializuje pole zveřejněných atributů (disclosed). Dále se zde naplní pole dostupných indexů pro ověřování v dané epoše, ve které existuje pouze 100 možností pro znáhodnění pseudonymu. Kdyby tato funkcionalita nebyla implementována a indexy by se volily náhodně, mohlo by se stát, že by se uživatel v jedné epoše prokázal vícekrát stejným pseudonymem, čímž by byla porušena nespojitelnosti relací. Posledním krokem této funkce je inicializace atributů.

Další funkcí je získání revokačního handleru od revokační autority, viz výpisek 3.3.

```
1  this.getRevocationAttributes = function (ra) {
2      this.revocationCred = ra.issue(this.id);
3      this.attributesFr[0] = this.revocationCred.m_r;
4      }
```

Výpis 3.3: Definice funkce getRevocationAttributes

Funkce pro získáni pověření od vydavatele je obdobná jako funkce na výpisku 3.3.

Nejkomplexnější funkcí, kterou třída User implementuje, je funkce `Show()`. Tato funkce implementuje všechny výpočty uživatelovy strany z obrázku 1.10. Celá funkce má více než sto řádků, z tohoto důvodu je na výpisku 3.4 představena jen vybraná část funkce.

```
let t_verify1 = mcl.mul(g1, ro_v);
let t_verify2 = new mcl.G1();
for (let i = 0; i < this.disclosed.length; i++) {
    if (!this.disclosed[i]) {
        t_verify2 = mcl.add(t_verify2, mcl.mul(this.
            auxilaryValues[i], randomizationArray[i]));
    }
}
let t_verify = mcl.add(t_verify1, mcl.mul(t_verify2, ro));
```

Výpis 3.4: Část funkce show()

Část funkce na výpisku 3.4 je ekvivalentní s výrazem $t_{\text{verify}} = g_1^{\rho_v} \sigma_{x_r}^{\rho_{mr} \cdot \rho} \prod_{z \notin D} \sigma_{x_z}^{\rho_{mz} \cdot \rho}$.

### 3.1.2 Třída Issuer

Jelikož základním předpokladem schématu RKVAC je, že vydavatel a ověřovatel jsou stejná entita, třída Issuer také zahrnuje všechny funkce týkající se vydavatele i ověřovatele, tedy funkce `Setup`, `Issue` a `Verify`. Funkce `Setup` vygenerejue soukromý klíč $sk_I = (x_0, \ldots, x_{n-1}, x_r)$. Implementace funkce Issue viz výpis 3.5.

```
this.Issue = function (attributesFr, id, sigma_ra) {
    //KONTROLA
    let hash = new mcl.Fr()
    hash.setHashOf(CryptoJS.SHA256(attributesFr[0].getStr
        (16) + id.getStr(16)).toString());
    if (!mcl.mul(mcl.pairing(sigma_ra, pk_ra), mcl.pairing
        (mcl.mul(sigma_ra, hash), g2)).isEqual(mcl.pairing(
        g1, g2))) {
         //podvrhnuty revokacni handler, konec protokolu
         return
    }//podepisovani
    let sum = this.secretKey;
    let auxilaryValues = [];
    for (let i = 0; i < attributesFr.length; i++) {
         let aux = mcl.mul(attributesFr[i], this.
             privateKeys[i]);
         sum = mcl.add(sum, aux);
```

```
14      }
15      let sigma;
16      sigma = mcl.mul(g1, mcl.inv(sum));
17      for (let i = 0; i < attributesFr.length; i++) {
18          let aux = mcl.mul(sigma, this.privateKeys[i]);
19          auxilaryValues.push(aux);
20      }
21      return [sigma, ...auxilaryValues];
22      }
```

Výpis 3.5: Funkce Issue()

Tato funkce zkontroluje, zda-li byl revokační handler podvržen. V případě, že byl, protokol ukončí bez vydání pověření. V opačném případě vypočte a vydá uživateli pověření $(\sigma, \sigma_{x_1}, \ldots, \sigma_{x_{n-1}}, \sigma_{x_r})$. Funkce `Verify` implementuje vydavatelovu část protokolu popsaného na obrázku 1.10. Z důvodu délky funkce, je prezentována výpisem 3.6 pouze část rekonstrukce $t_{\text{verify}}$.

```
let productNotDisclosed = new mcl.G1();
let productDisclosed = new mcl.G1();
for (let i = 0; i < disclosed.length; i++) {
    if (disclosed[i]) {
        productDisclosed = mcl.add(productDisclosed, mcl.
           mul(sigmaHat, mcl.mul(mcl.neg(e), mcl.mul(this.
           privateKeys[i], attributyFr[i]))));
        }
    else {
        productNotDisclosed = mcl.add(productNotDisclosed,
            mcl.mul(sigmaHat, mcl.mul(this.privateKeys[i],
            s_m[i])))
        }
}
let a = mcl.mul(sigmaHat, mcl.mul(mcl.neg(e), this.
   secretKey));
let b = mcl.mul(g1, s_v);
let c = productNotDisclosed;
let d = productDisclosed;
let t_1 = mcl.add(a, b);
let t_2 = mcl.add(c, d);
let t_verify = mcl.add(t_1, t_2);
```

Výpis 3.6: Funkce Verify()

Kód na výpisu 3.6 odpovídá výrazu $t_{\text{verify}} = \hat{\sigma}^{ex_0} \cdot g_1^{s_v} \cdot \prod_{z \notin D} \hat{\sigma}^{x_z s_{m_z}} \cdot \prod_{z \in D} \hat{\sigma}^{-ex_z m_z}$.

### 3.1.3 Třída Revocation Authority

Tato třída implementuje funkce `Setup`, `Issue`, `Revoke` a `Identify`. Funkce `Setup` vygeneruje pár soukromého a příslušného veřejného klíče tedy $sk_{\text{RA}} \in_R \mathbb{Z}_q, pk_{\text{RA}} = g_2^{sk_{\text{RA}}}$, poté inicializuje veřejnou revokační listinu a své dva soukromé revokační seznamy. V prvním z nich jsou uloženy informace, který revokační handler odpovídá kterému ID. Druhý mapuje vztah revokačního handleru s možnými pseudonymy korespondujícími s tímto handlerem v dané epoše.

Funkce `Issue` slouží k vydání revokačního handleru uživateli. Při tomto procesu také uloží uživatelovo ID s jeho revokačním handlerem do svého seznamu, tím vzniká možnost uživatele identifikovat. A také uloží jeho revokační handler spolu s všemi pseudonymy, které může v dané epoše použít do druhého ze svých seznamů.

```
1  this.identifyUser = function(c, epoch) {
2    let found_mr;
3    if (this.RD.has(epoch.toString())) {
4      let curr_epoch_rd = this.RD.get(epoch.toString());
5      for (let [m_r, c_i] of curr_epoch_rd) {
6        c_i.forEach((element) => {
7          if (element === c.getStr(16)) {
8            found_mr = m_r;
9          }
10       });
11     }
12
13   for (let [id, m_r] of this.RH) {
14     if (m_r === found_mr) {
15       return parseInt(id, 16).toString();
16     }
17   }
18 };
```

Výpis 3.7: Funkce Identify()

Funkce `Identify`, zobrazena na výpisu 3.7, nejprve najde svůj revokační seznam odpovídající přijaté epoše, ve kterém má uloženy dvojice revokačního handleru a všech možností psedudonymu odpoviídajících tomuto handleru v dané epoše. Tedy jeden handler a k němu patřících sto pseudonymů. V tomto seznamu na základě pseudonymu $C$, najde odpovídající revokační handler $m_r$. Poté v seznamu přidělených handlerů vyhledá na základě $m_r$ odpovídající ID uživatele, kterému byl tento $m_r$ přidělen.

## 3.2 Uživatelské rozhraní

Hlavní části webové aplikace frameworku Vue.js se nachází v souborech `App.vue` a `main.js`, které jsou automaticky vygenerované při vytváření nového projektu. V souboru `main.js` se nachází inicializace instance Vue. V tomto souboru se také provádí import všech pluginů, které chceme v aplikaci použít. Tedy Vuetify, BootstrapVue, knihovny MCL a knihovny CryptoJS. Uživatelské rozhraní využívá pro výpočty objekty definované v sekci 3.1.

Na výpisu 3.8 je příklad volání funkce kryptografického jádra z uživatelského prostředí. Konkrétně funkce `Issue()`, kterou spustí tlačítko "Sign my attributes! ". Tato funkce vygeneruje uživateli náhodné ID v rozsahu 0-100, uživateli vydá revokační handler od revokační autority a poté pověření od vydavatele. Následně změní uživateli aktivní záložku na "Show-Verify".

```
1  issue(){
2      this.uniqueID = Math.floor(Math.random() * Math.floor
           (100));
3      this.uzivatel.id.setStr(this.uniqueID.toString());
4      this.uzivatel.getRevocationAttributes(this.ra);
5      this.uzivatel.getExactAttributes(
6          this.vydavatel,
7          this.name,
8          this.selectPozice,
9          this.email,
10     );
11     this.tabIndex = 2;
12     },
```

Výpis 3.8: Volání funkce Issue

Díky Vue a Bootstrapu je možné jednoduše vytvářet interaktivní tlačítka. Příklad definice tlačítka je na výpisu 3.9.

```
1  <b-button
2    block
3    :class="visibleIssue ? null : 'collapsed'"
4    :aria-expanded="visibleIssue ? 'true' : 'false'"
5    aria-controls="collapse-issue"
6    @click="visibleIssue = !visibleIssue"
7  >
```

Výpis 3.9: Příklad tlačítka

Tlačítko obsahuje hned několik parametrů závislých na dynamicky se měnícím obsahu. První z nich v závislosti na proměnné `visibleIssue` rozbaluje nebo naopak zabaluje prvek, jmenovaný parametrem `aria-controls`, tedy `collapse-issue`. Posledním parametrem je `@click`, který udává, co se stane po kliknutí na tlačítko. V tomto případě se pravdivostní hodnota proměnné `visibleIssue` změní na její opak.

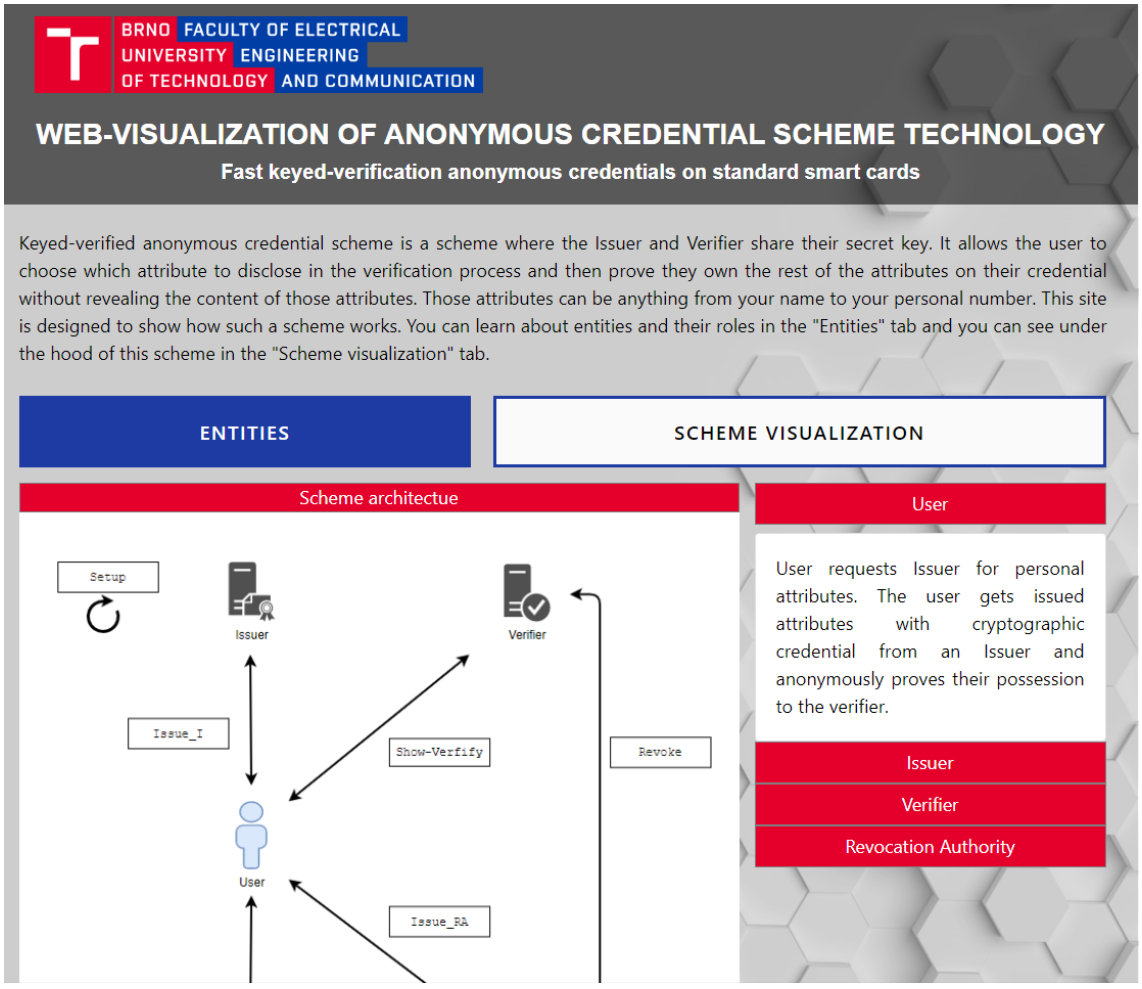
```
1 <v-form ref="form" v-model="valid" class="pl-3 pr-4">
2             <v-text-field
3               v-model="name"
4               :rules="[(v) => !!v || 'Name is required']"
5               label="Name"
6               required
7             ></v-text-field>
```

Výpis 3.10: Příklad textového pole formuláře

# 4 Webová aplikace

V této kapitole je demonstrována funkčnost aplikace jako celku. Cílem práce je vytvořit uživatelsky přívětivou webovou aplikaci, znázorňující průběh a možnosti uplatnění protokolu RKVAC. V této aplikaci by měl uživatel zjistit, jak schéma funguje, k čemu slouží, jaké entity v něm vystupují a jakých protokolů využívá.

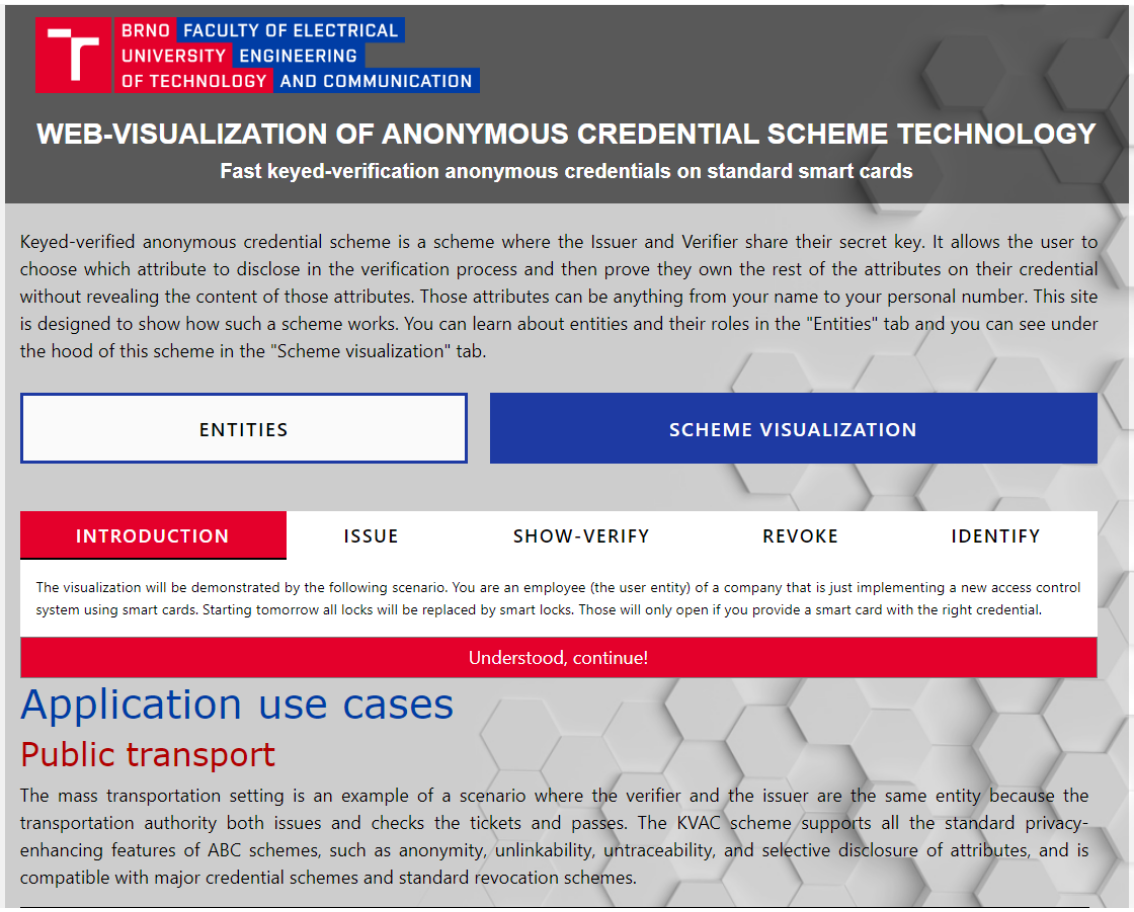
V hlavní části webové stránky, viz obrázek 4.1, se nachází krátký popis schématu a dvě tlačítka. První z nich, označeno Entities, uživatele přesměruje na stránku popisující architekturu schématu a entity v něm vystupující. Druhé, označeno Scheme Visualization, pak na stránku, kde se nachází interaktivní příběh provázející uživatele schématem RKVAC.




Obr. 4.1: Hlavní stránka aplikace

Na stránce entities se v levé části nachází obrázek znázorňující architekturu schématu. Jsou na něm zobrazeny jednotlivé entity a protokoly, které mezi nimi mohou probíhat. V pravé části se nachází tlačítka, která se po kliknutí rozbalí a zobrazí kartu s krátkým popisem jednotlivých entit. Obsah stránky scheme visualization je

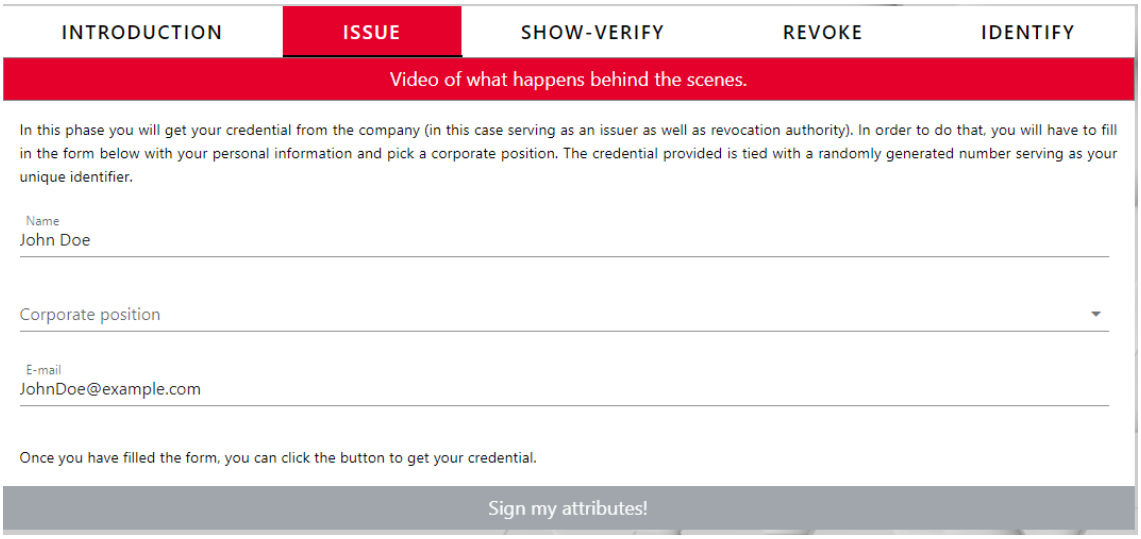
poté rozdělen do záložek, viz obrázek 4.2, které uživatele provedou protokolem. Od úvodu, který uživateli vysvětlí, co se před ním vlastně nachází, přes vydání pověření až po prokázání se ověřovateli. Poté následují volitelné záložky, ve kterých si uživatel může vyzkoušet revokaci svého pověření, nebo odhalení své identity.




Obr. 4.2: Stránka Scheme Visualization

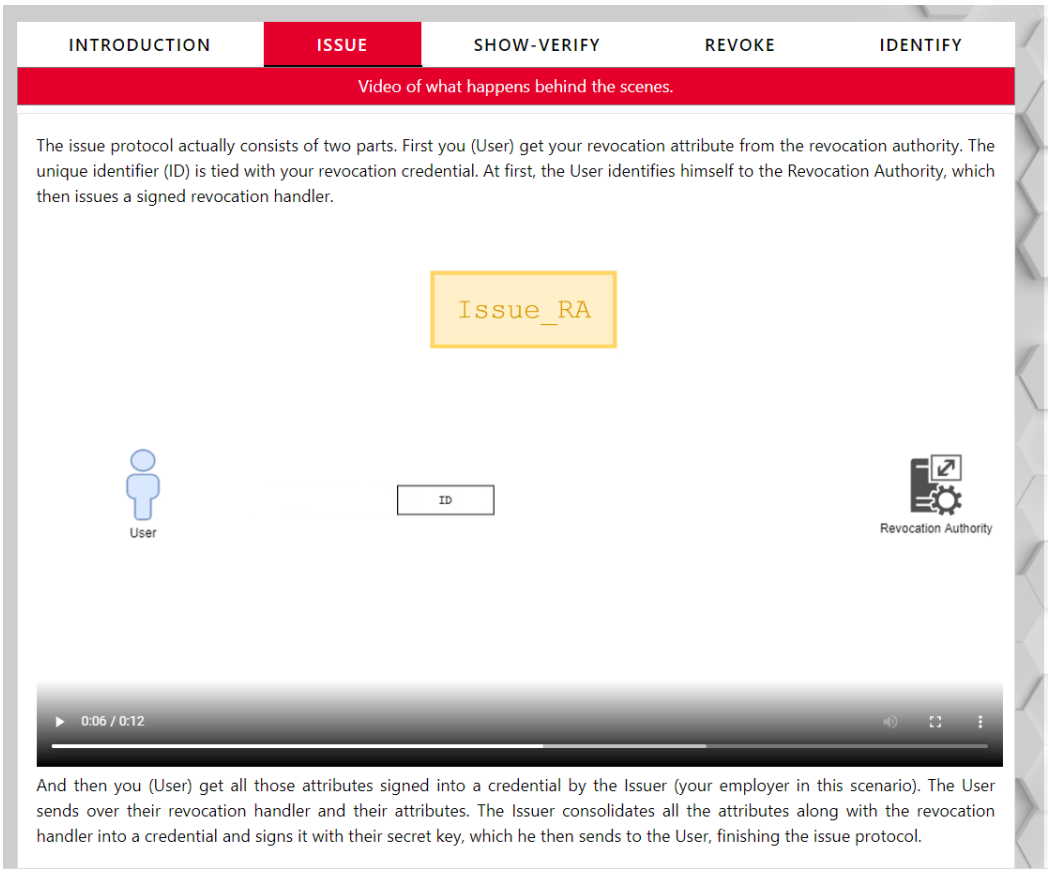
V záložce Introduction, kterou je možné vidět na obrázku 4.2, je uživateli vysvětleno, že v rámci demonstrace funkcionality schématu RKVAC bude hrát roli zaměstnance firmy, která nově implementuje systém chytrého řízení přístupu v budově firmy. Každý zaměstnanec bude ke vstupu do jednotlivých místností potřebovat smart kartu, aby u dveří prokázal, že má povolení jimi projít.

V záložce Issue, viz obrázek 4.3, je uživateli představen formulář, ve kterém si může zvolit své osobní atributy. Po správném vyplnění formuláře se tlačítko “Sign my attributes! “ aktivuje a je možné na něj kliknout. Kliknutím na tlačítko je uživateli vydán revokační handler a následně i odpovídající pověření.



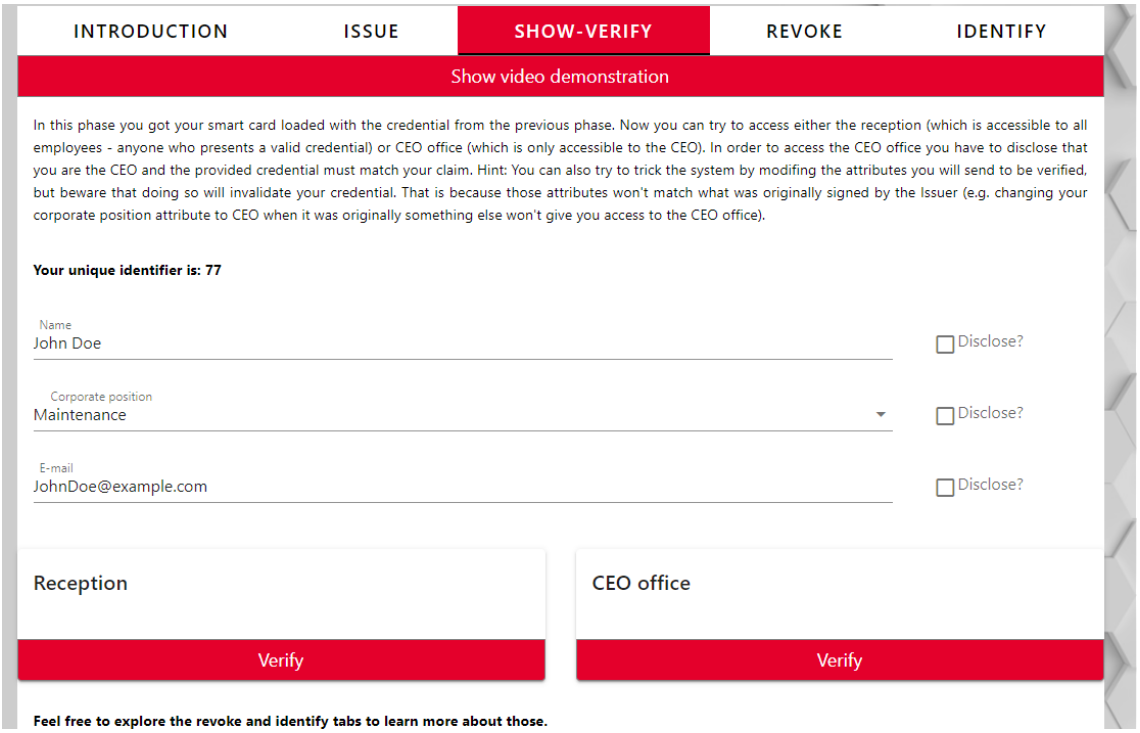

Obr. 4.3: Záložka Issue

Uživatel má dále možnost stisknout tlačítko "Video of what happens behind the scenes.", které se po stisknutí rozbalí a uživateli zpřístupní video-demonstraci vydávacího protokolu.




Obr. 4.4: Video-demonstrace v záložce Issue

V záložce Show-Verify má uživatel možnost vyzkoušet si ověřovací protokol. V této fázi si uživatel může vybrat ze dvou místností, recepce a kanceláře ředitele, do které se pokusí vstoupit. V případě, že se rozhodne vstoupit do recepce, je podmínkou ke vstupu držení platného pověření jakéhokoliv zaměstnance firmy. V případě, že se pokusí vstoupit do kanceláře ředitele firmy, musí prokázat držení atributu "CEO", neboli ředitel. To tedy znamená, že ačkoliv má uživatel platné pověření s atributem "CEO", ale rozhodne se, že tento atribut nezveřejní, dveře se neotevřou. Také platí, že pokud uživatel využije možnosti změnit svůj atribut, například pokud se rozhodne vydávat za ředitele, i když jim není, systém jeho žádost o ověření zamítne a uživateli se objeví zpráva, že se pravděpodobně pokoušel falšovat atributy. Strukturu záložky Show-Verify je možno vidět na obrázku 4.5.




Obr. 4.5: Záložka Show-Verify

V záložce Revoke je uživatel seznámen se skutečností, že dle příběhu ztratil peněženku i se svou smart kartou. Je proto nutné pověření revokovat a zabránit jeho zneužití. Po kliknutí na tlačítko Revoke, je uživateli revokováno pověření a všechny další pokusy o ověření v záložce Show-Verify budou zamítnuty a uživateli se objeví zpráva, že bylo jeho pověření revokováno.

V záložce Identify je uživatel seznámen s možností odhalení identity útočníka za pomocí revokační autority. Dle příběhu se hledá osoba, která zničila vybavení kanceláře. Hlavním podezřelým je tedy osoba, která naposledy otevřela dveře této kanceláře. Jelikož ale systém poskytuje anonymizaci, je znám pouze pseudonym této

osoby. Na řadu tedy přichází revokační autorita, která je schopna pomocí pseudonymu a časového údaje odhalit identitu útočníka.

V každé z těchto záložek se nachází video, vizualizující průběh každého z protokolů.

## 4.1 Spuštění webové aplikace

Webová aplikace potřebuje pro správné zobrazení server, který bude uživateli na vyžádání posílat obsah této stránky. Pro tento účel je možno využít kteréhokoliv z dostupných řešení. Nejjednodušším řešením je zprovoznit modul "Simple Http Server" programovacího jazyka Python. Prvním krokem je tedy nainstalovat interpret jazyka Python, který je dostupný ve verzích pro Windows, Linux, MacOS a další. Ze stránky python.org stáheme instalátor pro příslušný systém a pokračujeme dle kroků instalace. Po dokončení instalace interpretu python je již možné se pomocí příkazové řádky dostat do složky `dist` příkazem

```
cd (cesta_k_priloze)/dist
```

Poté už zbývá jen zapnout server pomocí příkazu

```
python -m http.server 8000
```

Tento příkaz spustí server na lokální adrese (localhost, nebo 127.0.0.1) a portu 8000. Po spuštění serveru už je stránka dostupná z webového prohlížeče na adrese `localhost:8000`.

Systémové požadavky jsou minimální, avšak pro správné fungování webové aplikace je nutné mít v prohlížeči povolený JavaScript. Aplikace byla testována v prohlížečích Google Chrome verze 83, Microsoft Edge a Mozzila Firefox.

# Závěr

V teoretické části byla popsána kryptografická primitiva nezbytná k vytvoření zadané aplikace. Jsou zde vysvětleny základní pojmy spadající do dané problematiky, závazková schémata, protokoly s nulovou znalostí a představeno je i podpisové schéma weak Boneh-Boyen, na kterém je založena vytvořená aplikace.

Poslední kapitola byla věnována detailnímu popisu implementace a demonstraci funkčnosti vytvořené webové vizualizační aplikace.

V rámci bakalářské práce byla v souladu se zadáním navržena a implementována webová vizualizace anonymního pověřovacího schématu RKVAC zadaného vedoucím práce. Pro vývoj webové aplikace byly vyžity technologie HTML 5, JavaScript,css a knihovny MCL. Knihovna MCL byla zvolena na základě nejlepších výkonnostních testů ze všech analyzovaných knihoven podporujících bilineární párování.

Výsledná aplikace implementuje kryptografické jádro protokolu RKVAC, konkrétně protokoly Issue_I, Issue_RA, Show-Verify, Revoke a Identify.

Webová vizualizace umožňuje demonstraci funkcionality protokolu RKVAC pomocí předpřipravených videoukázek vytvořených pomocí MicroSoft Powerpoint. Aplikace dále umožňuje uživateli zadávat vstupy a tím demonstruje funkcionalitu a využitelnost schématu RKVAC v prostředí, se kterým je možné se setkat v životě.

# Literatura


[1] Šárka Chwastková. Atributová autentizace, 2018.

[2] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology — EUROCRYPT' 87*, pages 127–141, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[3] C. P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, 1989.

[4] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, February 1989. URL: `https://doi.org/10.1137/0218012`, `doi:10.1137/0218012`.

[5] Chin-Ling Chen, Tzay-Farn Shih, Yu-Ting Tsai, and De-Kui Li. A Bilinear Pairing-Based Dynamic Key Management and Authentication for Wireless Sensor Networks. *Journal of Sensors*, 2015:14, 2015. URL: `10.1155/2015/534657`.

[6] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, Apr 2008. URL: `https://doi.org/10.1007/s00145-007-9005-7`, `doi:10.1007/s00145-007-9005-7`.

[7] David Jao and Kayo Yoshida. Boneh-boyen signatures and the strong diffie-hellman problem. In Hovav Shacham and Brent Waters, editors, *Pairing-Based Cryptography – Pairing 2009*, pages 1–16, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[8] Jan Camenisch, Manu Drijvers, Petr Dzurenda, and Jan Hajny. Fast keyed-verification anonymous credentials on standard smart cards. In Gurpreet Dhillon, Fredrik Karlsson, Karin Hedström, and André Zúquete, editors, *ICT Systems Security and Privacy Protection*, pages 286–298, Cham, 2019. Springer International Publishing.

[9] Jan Camenisch, Manu Drijvers, and Jan Hajny. Scalable revocation scheme for anonymous credentials based on n-times unlinkable proofs- in wpes '16 proceedings of the 2016 acm on workshop on privacy in the electronic society. pages 123–133, New York, NY, USA, 2016. Association for Computing Machinery.

[10] Dagmar Wikarská. Kryptografie na programovatelných čipových kartách, 2019.

[11] Html. URL: https://developer.mozilla.org/en-US/docs/Web/HTML.

[12] Js. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[13] Css. URL: https://developer.mozilla.org/en-US/docs/Web/CSS.

[14] Shigeo Mitsunari. Mcl. URL: https://github.com/herumi/mcl.

[15] Petr Dzurenda. Cryptographic protection of digital identity, 2019.

[16] Vuetify. URL: https://vuetifyjs.com/en/.

[17] Christian Paquin. U-prove technology overview v1.1 (revision 2), April 2013. URL: https://www.microsoft.com/en-us/research/publication/u-prove-technology-overview-v1-1-revision-2/.

[18] Visual studio code. URL: https://code.visualstudio.com/.

[19] Marek Sikora. Webová vizualizace kryptografických systémů, 2015.

# Seznam symbolů, veličin a zkratek

| | |
|---|---|
| **wBB** | podpis weak Bohen-Boyen |
| **SDH** | Strong Diffie-Hellman |
| **MAC** | Message authentication code |
| **RKVAC** | Revocable Keyed-Verification Anonymous Credential scheme |
| **DOM** | Document Object Model |
| **RSA** | Rivest, Shamir, Adleman |
| **npm** | Node package manager |
| **HTML** | HyperText Markup Language |

# Seznam příloh